METHOD AND AGENT FOR THE PROTECTION AGAINST THE UNAUTHORISED USE OF COMPUTER RESOURCES

Field of the Invention

The present invention relates to the security management of computers. More particularly, the invention relates to a method and an agent for preventing the access to the use of computer resources by hostile applications.

Background of the Invention

The Internet has developed very much both in respect of its contents and of the technology employed, since it began a few years ago. In the early days of the Internet, web sites included text only, and after a while graphics was introduced. As the Internet developed, many compressed standards, such as pictures, voice and video files, were developed and with them programs used to play them (called "players"). Initially, such files were downloaded to the user's workstation only upon his request, and extracted only by the appropriate player, and after a specific order from the user.

When, in the natural course of the development of the World Wide Web the search for a way to show nicer, interactive and animated Web Pages began, Sun Microsystems Inc. developed Java - a language that allows the webmaster to write a program, a list of commands - Network Executables - that will be downloaded to the user workstation most of the time without his knowledge, and executed by his browser at his workstation. The executables are used, e.g., to provide photographic animation and other graphics on the screen of the web surfer. Such executables have ways of approaching the user

workstation's resources, which lead to a great security problem. Although some levels of security were defined in the Java language, it was very soon that a huge security hole was found in the language.

Since Java was developed, Microsoft developed ActiveX, which is another Network Executable format, also downloaded into the workstation. ActiveX has also security problems of the same kind.

The Internet has been flooded with "Network Executables" which may be downloaded -- deliberately or without the knowledge of the users -- into workstations within organizations. These codes generally contain harmless functions. Although usually safe, they may not meet the required security policy of the organization.

Once executed, codes may jam the network, cause considerable irreversible damage to the local database, workstations and servers, or result in unauthorized retrieval of information from the servers/workstations. Such elements may appear on Java applets, ActiveX components, DLLs and other object codes, and their use is increasing at an unparalleled pace. The majority of these small programs are downloaded into the organization unsolicited and uncontrolled. The enterprise has no way of knowing about their existence or execution and there is no system in place for early detection and prevention of the codes from being executed.

The problem is made worse, in some cases, by the existence of large intranets and LANs, which may also be used by unauthorized persons to access workstations and

perform hostile activities thereon.

The security problem was solved partially by the browser manufactures which allow the user to disable the use of executables. Of course this is not a reasonable solution, since all the electronic commerce and advertising are based on the use of executables.

In three copending patent applications of the same applicants hereof, IL 120420, filed March 10, 1997, IL 121815, filed September 22, 1997, and IL 122314, filed November 27, 1997, the descriptions of which are incorporated herein by reference, there are described methods and means for preventing undesirable Executable Objects from infiltrating the LAN/WAN in which we work and, ultimately, our workstation and server. IL 122314 further provides a method for enforcing a security policy for selectively preventing the downloading and execution of undesired Executable Objects in an individual workstation.

While much has been done in the abovementioned patent applications toward protecting the individual workstation, one problem yet remained unsolved: the hostile use of local resources by applications which have passed any earlier security check (e.g., a gateway security policy), because they did not contravene such security policy, or by applications which have not passed through an earlier check point (such as a gateway equipped with a security policy check, as described in the aforementioned Israeli patent applications), either because such earlier point of check is not available, or because the application has been loaded directly on the workstation. Such hostile use of CPU resources may lead to damage to the data, operation and hardware of the workstation and, under the conditions

contemplated above, may go undetected until the damage is done.

It is an object of the present invention to provide a method and agent which overcomes the aforesaid drawbacks of prior art methods, and which provides effective protection at the workstation level.

It is another object of the present invention to provide a method and an agent which can be used effectively to prevent the hostile use of workstation resources by applications running on said workstation.

Other objects and advantages of the invention will become apparent as the description proceeds.

SUMMARY OF THE INVENTION

In one aspect, the invention is directed to a method for preventing an hostile use of computer resources by an application running on a workstation, comprising the steps of:

a) providing a list of services that are not allowed for access by unspecified applications;

b) when such unspecified application runs on the workstation, preventing said application from accessing any resource directly;

c) analyzing any direct or indirect request for access to specific services, to determine whether such request is allowable according to the list defined under a) above;

d) if the request is allowable, allowing the workstation to process it; and

e) if the request is not allowable, preventing the unspecified application from accessing the requested resource;

wherein said resource may be any local or remote resource, including, but not limited to, memory allocation, files, directories, operations with files and directories, such as copy, delete or compress, or any other operation leading to a change in the workstation or its periphery. Illustrative - but not limitative - examples of such operations include access to system files, configuration information, network communications, hardware equipment (floppy, modem, etc.), CMOS data (time, date, etc.), or the use of resources such as memory allocation, process creation, threads creation, use of excessive CPU time, use of excessive disk space, use of excessive network communication, and use of excessive graphical resources and use of system or application configuration.

According to a preferred embodiment of the invention the list of services is provided as a look-up table.

By "unspecified application" it is meant to indicate an application that is not specifically identified in a pre-set list of applications. According to a preferred embodiment of the invention, said pre-set list of applications includes a list of resources which each application may utilize.

In another aspect, the invention is directed to an agent for protecting a workstation against the hostile use of computer resources by an unspecified application running on

said workstation, comprising:

a) means for detecting an unspecified application or a module of an application running on the workstation;

b) means for determining the requests for resources to be used by said unspecified application;

c) means for identifying chain requests for resources utilization, wherein said chain requests comprise requests made by resources called by said unspecified application;

d) means for determining whether requests made directly by said unspecified application are allowable;

e) means for determining whether requests made indirectly, as chain requests, by said unspecified application would be not allowable if made directly by said unspecified application; and

f) means for preventing said chain request from being processed, if it is determined that the request is not allowable, or that it would not be allowable if made directly by said unspecified application, and for allowing its processing if otherwise determined.

According to a preferred embodiment of the invention, the means for determining whether requests made directly or indirectly by said unspecified application are allowable comprise a look-up table including a list of services that are not allowed for access by unspecified applications. In another preferred embodiment of the invention, the agent comprises a pre-set list of applications including a list of resources that each

application may utilize.

All the above and many other characteristics and advantages of the invention, will be better understood through the following illustrative and non-limitative examples of preferred embodiments thereof, with reference to the appended drawings.

Brief Description of the Drawings

Fig. 1 schematically illustrates different applications and their requests and related operations;

Fig. 2 schematically illustrates a detail of an illustrative application that will cause machine malfunctioning; and

Fig. 3 illustrates a situation in which indirect unallowable resource exploitation is attempted.

Detailed Description of Preferred Embodiments

Examples of such situations are exemplified in Figs. 1-3. Referring to Fig. 1, three different applications are shown, marked APP1 through APP3. The process takes place at three different levels: the user mode (indicated by "U.M."), the kernel mode (indicated by "K.M."), and the hardware (indicated by "H.W."). The three different modes are schematically separated in the figure by straight lines. The APP1, APP2 and APP3 applications operate in the user mode. APP1 is an "open file" I/O request. This request is passed on to the I/O manager, which, in turn, refers to the disk(s) to perform the required operation. A filter (indicated as "S7 Filter" in the figure) analyzes the

request to determine whether it is permissible according to the security policy. If it is permissible, it is allowed to proceed to the I/O manager, which processes the request with the disk(s).

APP2, on the other hand, makes a request involving the network, i.e., and "open connection to the file server" request. The network manager is allowed to process this request only if the filter S7 has determined that it is permissible. Similarly, APP3 makes a memory allocation request, which is examined by the filter and, if permissible, is passed on to the memory manager and then acted upon in connection with the memory.

The operation of the various requests in the kernel mode and *vis-a-vis* the hardware, after the filter has examined and allowed them, is the same as with conventional operations in everyday computer, is well known to the skilled person, and therefore is not described herein in detail, for the sake of brevity.

Looking now at Fig. 2, a detail of an illustrative application that will cause machine malfunctioning is shown. In this example APP1 generates 1000 requests to generate new processes. If the system of the invention is not present, the 1000 requests will be passed on to the CPU by the Process Manager, and will use all the resources of the CPU, thus holding the work of the machine. If the filter of the invention is present, however, it may be pre-set to allow the generation of only a limited number of processes by the same application. Therefore, if a number of new processes are requested by a single application, which exceeds the preset limit, the filter S7 will not allow it to pass on to the process manager, thus avoiding the exhaustion of the resources of the machine.

Fig. 3 illustrates a situation in which indirect unallowable resources exploitation is attempted. In this example APP1 is of a type that is not allowed to send a request to the I/O Manager. If it attempts to do so, it is stopped by the S7 Filter, unless the request complies with the Security Policy preset with S7. APP1 may therefore be programmed so as to effect an interprocess communication, viz., to communicate its request to a further process, APPX, which is permitted to make the request that APP1 is not allowed to make, to the I/O/ Manager. In this case, the S7 filter between the User Mode and the Kernel Mode is bypassed. In order to prevent such an occurrence, a further filter S7 is located between all communicating processes, and stops any request that is passed on to one process to the other (in the example, from APP1 to APPX), and which the first process is not allowed to make directly.

Of course, as will be apparent to the skilled person, the filter S7 is not a physical filter, but rather a logical one. Logical filters of this kind can be provided in a plurality of ways, using many different analysis processes and criteria, which will be predetermined by the skilled person according to the particular requirements of the system involved.

All the above description and examples have therefore been provided for the purpose of illustration only, and are not intended to limit the invention in any way, except as defined by the appended claims.